Client Web Services Frameworks

Session 804

Client Web Services Frameworks

Steve Zellers
Web Services Engineering

What Is a Web Service?

- Accessing an ‘endpoint’ or method over the web
- Parameters are written as XML; transported over HTTP (usually)
- Heterogeneous—different platforms and object models
- Commercial and Private
- Simple examples:
    - BarnesAndNoble.com price of book by ISBN
    - Current temperature in Cupertino

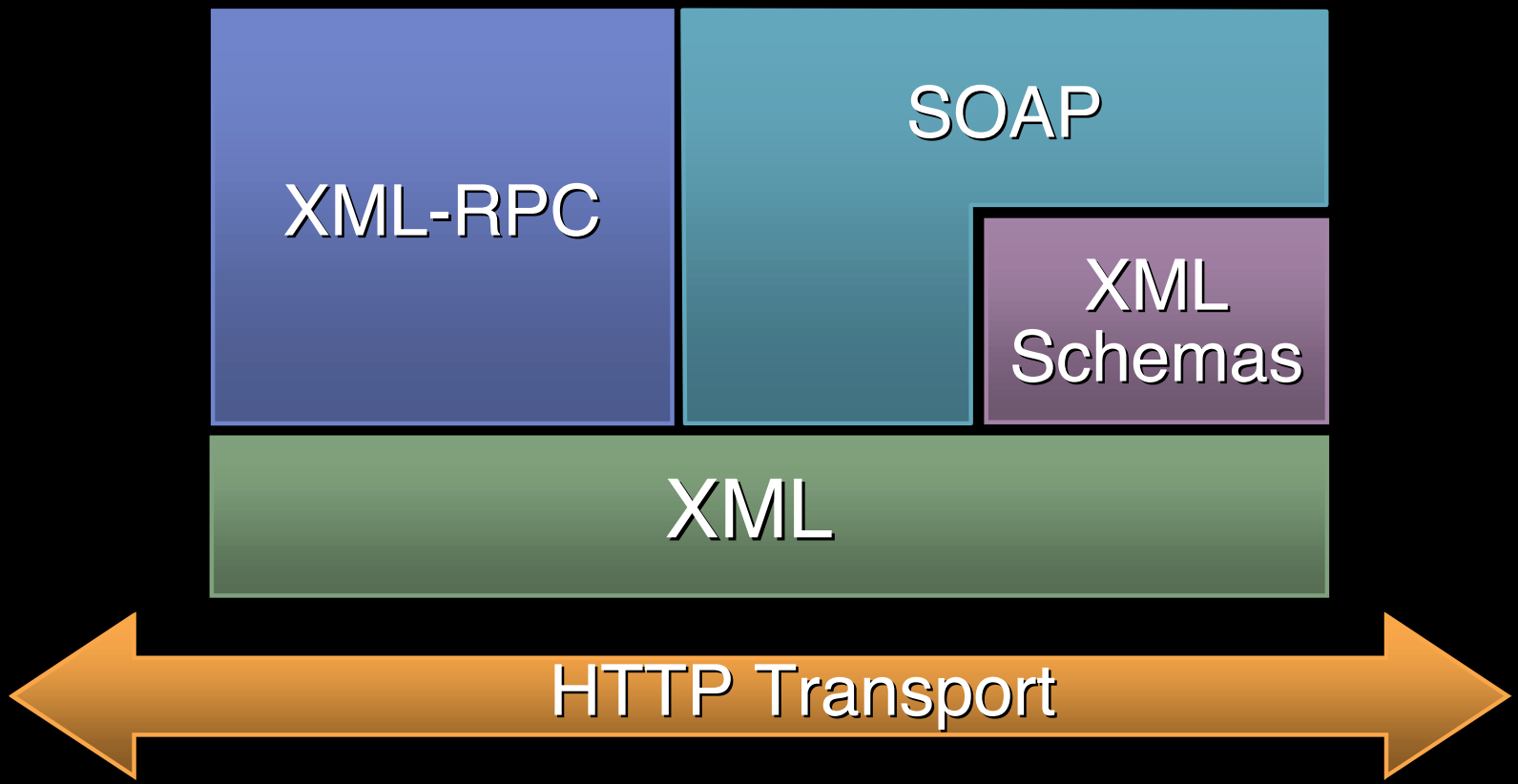
XML Standards: Acronym City!

- XML-RPC, SOAP
- XML, XSLT, XML Schemas
- XPATH, SAX, DOM
- HTTP, SMTP, DIME, FTP
- WSDL, UDDI
- WS-I, W3C

XML Standards: Acronym City!

- XML-RPC, SOAP
- XML, XSLT, XML Schemas
- XPATH, SAX, DOM
- HTTP, SMTP, DIME, FTP
- WSDL, UDDI
- WS-I, W3C

Web Services: Building a Standard



State of XML Web Services

- XML-RPC
  - UserLand, independent developers
  - “Done”
- SOAP 1.0, 1.1, 1.2
  - Microsoft, UserLand, DevelopMentor, IBM, Lotus
  - Extensions spearheaded by MS, IBM
  - 1.1 is a w3c recommendation (“done”)
  - 1.2 is a w3c “Note” (not “done”)

XML-RPC vs. SOAP

- XML-RPC
  - Very simple
  - Basic type support (scalar, array, record)
  - Positional parameters
  - No international text in <string> elements
  - HTTP only transport
- SOAP (1.1)
  - Air of simplicity
  - Enhanced type support
    - MultiRef
    - User types
  - Prepositional (named) parameters
  - Transport agnostic

Accessing Web Services

- There are 50+ XML-RPC implementations, 80+ SOAP implementations
- Implementations bind a language runtime to a serialization format
- It's an investment to bring in one of these toolkits . . .
- Choose the toolkit that matches your object model (perl, python, Java, Cocoa, C, AS)

Mac OS X 10.1: Baked in Support

- SOAP 1.1 and XML-RPC support via AppleEvents
- AppleScript supported ‘for free’
- Very popular high-level way to access corporate and public web services
- Hijacks typeApplicationURL addressing mode
- Automatically works with common AppleEvent types
- No API—entirely data driven

AE Sample: XML-RPC Over AE

- "What is state number 41, alphabetically?"
- Server: **http://betty.userland.com:80/RPC2**
- Method: **examples.getStateName**
- Parameters: **a single integer**
- Returns: **a string**

Message/Response

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param>
            <value>
                <i4>41</i4>
            </value>
        </param>
    </params>
</methodCall>
```

Message/Response

```
<?xml version="1.0"?>
<methodResponse>
   <params>
         <param>
               <value>South Dakota</value>
         </param>
   </params>
</methodResponse>
```

AE Sample—Just AppleEvents:

- Create parameter list (a single integer)
- Create the direct object
- Create the event
- Send it and deal with the reply

AE Sample—Just AppleEvents:

- Create parameter list (a single integer)
- Create the direct object
- Create the event
- Send it and deal with the reply

AE Sample—
Create the Parameter List

```
AEDesc paramList;
AECreateList(NULL, 0, true, &paramList);

SInt32 ixState = 41;
AEPutPtr(&paramList, 0, typeSInt32, &ixState,
sizeof(ixState));
```

It Is Just AppleEvents:

- Create parameter list
- Create the direct object
- Create the event
- Send it and deal with the reply

AE Sample—
Create the Direct Object

```
// Direct object is a record
AEDesc directObject;
AECreateList(NULL, 0, true, &directObject);

// With two fields:
AEPutParamDesc(&directObject,
   keyRPCMethodParam, &paramList)
AEPutParamPtr(&directObject,
   keyRPCMethodName, typeChar,
   “examples.getStateName”, strlen(…))
```

AE Sample—
Create the Direct Object

```
// Direct object is a record
AEDesc directObject;
AECreateList(NULL, 0, true, &directObject);

// With two fields:
AEPutParamDesc(&directObject,
   keyRPCMethodParam, &paramList)
AEPutParamPtr(&directObject,
   keyRPCMethodName, typeChar,
   “examples.getStateName”, strlen(…))
```

It Is Just AppleEvents:

- Create parameter list
- Create the direct object
- Create the event
- Send it and deal with the reply

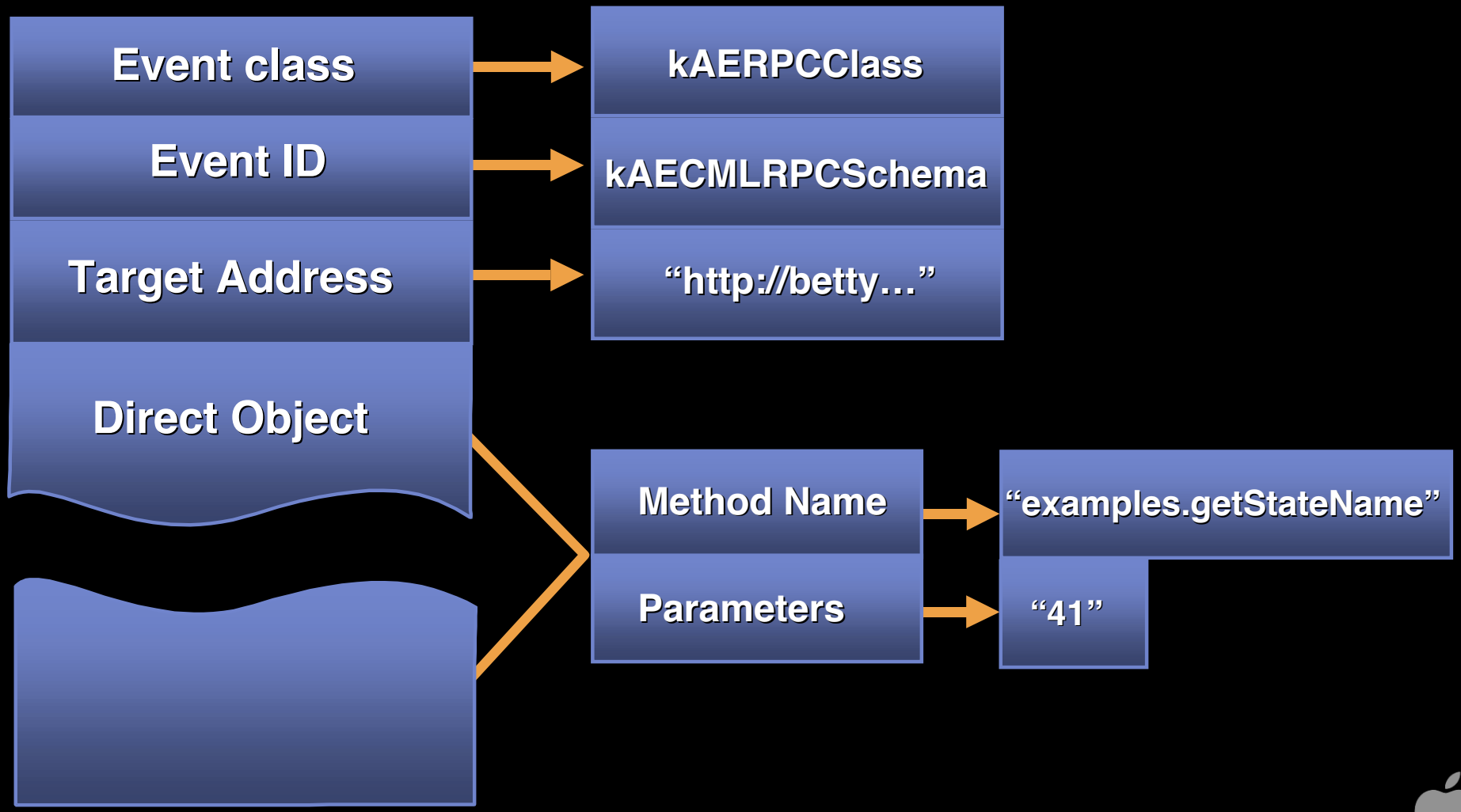
AE Sample—Create the Event

```
// Describe the address (endpoint)
AEDesc addr;
AECreateDesc(typeApplicationURL,
"http://betty.userland.com/RPC2", strlen(…), &addr)

// Build the event
AEDesc event;
AECreateAppleEvent(kAERPCClass, kAEXMLRPCScheme,
&addr, …, &event);
AEPutParamDesc(&event, keyDirectObject, &directObject)
```

AE Sample—Create the Event

```
// Describe the address (endpoint)
AEDesc addr;
AECreateDesc(typeApplicationURL,
"http://betty.userland.com/RPC2", strlen(…), &addr)

// Build the event
AEDesc event;
AECreateAppleEvent(kAERPCClass, kAEXMLRPCScheme,
&addr, …, &event);
AEPutParamDesc(&event, keyDirectObject, &directObject)
```

AE Sample—Create the Event



It Is Just AppleEvents:

- Create parameter list
- Create the direct object
- Create the event
- Send it and deal with the reply

AE Sample—Send the Event

```
AEDesc reply;
AESend(&event, &reply, kAEWaitReply, …, …)

AEGetParamPtr(&reply, keyDirectObject, typeChar,
    buffer, sizeof(buffer), …, &actualSize)
printf(“State: %.*s\n”, actualSize, buffer);
```

More Complex Types

- **AEList** an ordered **<array>**
- **AERecord** an unordered **<struct>**
- But: **AERecord** uses four character keys!
- **keyASUserRecordFields** to the rescue:
  - Create an **AERecord** with a single **AEList** with this key
  - The **AEList** is **{ name, value, … }**

Example: Creating a Fancy AERecord

AEDesc list; AECreateList(NULL, 0, false, &list)

AEDesc record; AECreateList(NULL, 0, true, &record)

AEPutPtr(&list, 0, “myKey”, strlen())

AEPutPtr(&list, 0, “some value”, strlen())

AEPutParamDesc(&record, keyASUserRecordFields, &list)

AppleScript: An Alternative

- Stub dictionary provides ‘call soap’ and ‘call xmlrpc’
- Great way to prototype and debug web services

```
tell application “http://betty.userland.com:80/RPC2”
        call xmlrpc { method name:
        “examples.getStateName”, parameters: { 41 } }
end
```

- Very easy to embed AppleScript/OSA in applications

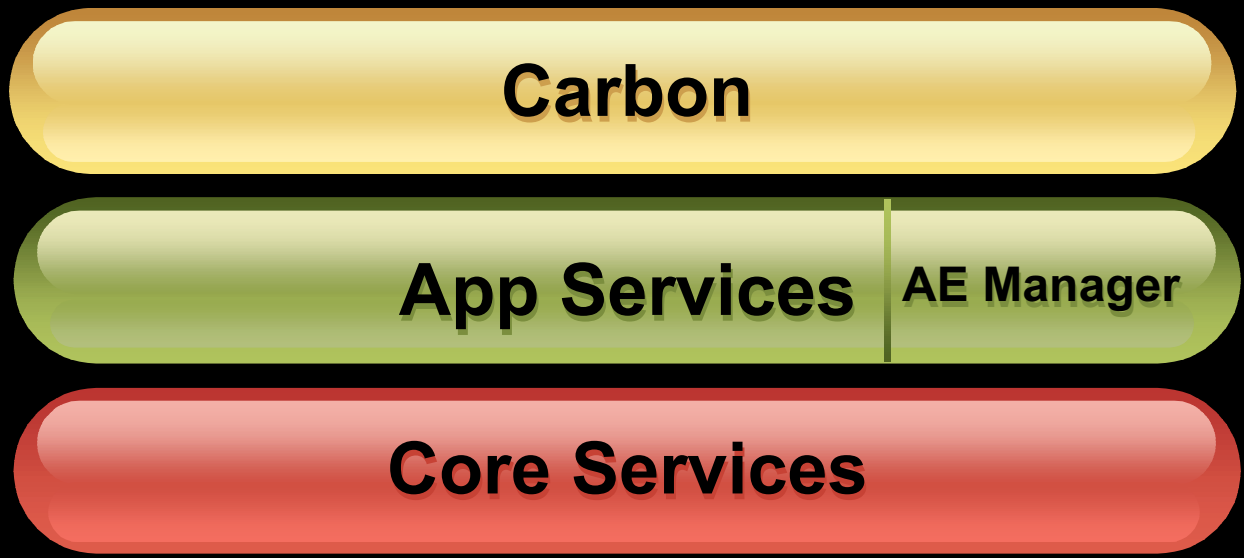
Demo

Accessing a Web Service With AppleScript Studio

Tim Bumgarner
AppleScript Studio Engineer

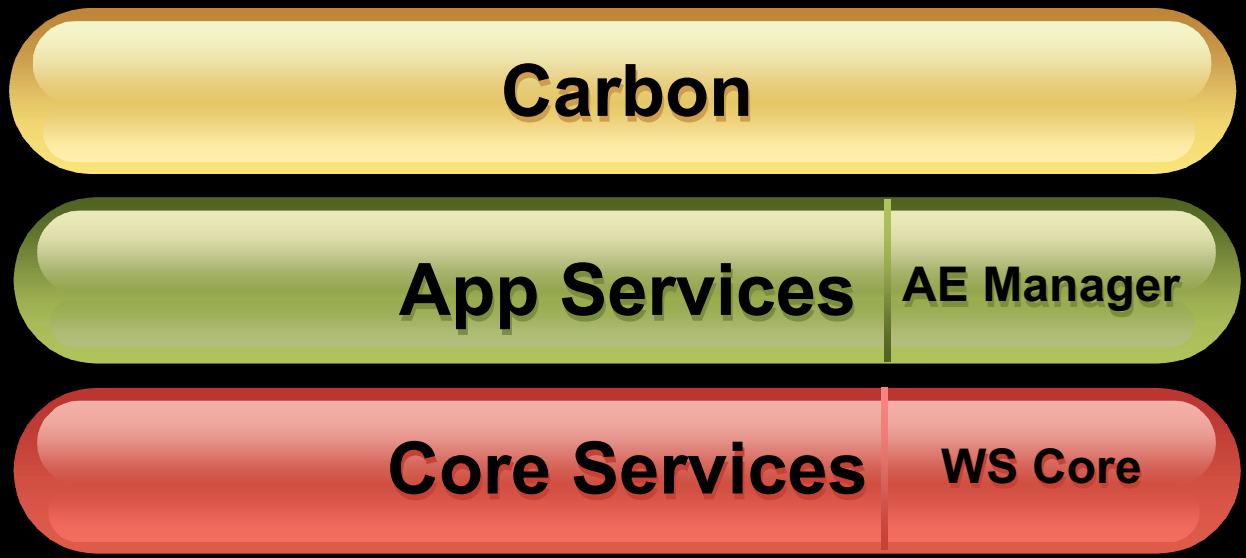
Announcing: WebServicesCore

- What? Another one? What was wrong with the AppleEvent method?



Announcing: WebServicesCore

- Subframework of CoreServices umbrella—available to dœmons as well as applications, plug-ins, tools, etc.



WS: Integration

- Leverages CFXMLParser, CFNetwork
- Thread safe, RunLoop Safe and Friendly
- CFType based—toll free bridging with ObjC
- Simple procedural API

WS: Features

- The API is protocol independent—SOAP or XML-RPC
- Synchronous “one-shot” operation, or async with callbacks
- Based on advances in CFNetwork
    - Advanced networking options for firewall, proxies, IPV6 comes for free
    - Full access to the underlying CFHTTPMessage
- Conceptually similar to scheduling in CFStream
- HTTP / HTTPS support in 1.0
    - Other protocols will follow

WSMethodInvocationRef

- Created with endpoint url and method name
- Prototype parameter specifies encoding style (XML-RPC or SOAP v1.1)
- Properties control serialization rules and transport options
- Parameters described by CFTypes
- Schedule on a RunLoop—or call Synchronously

WSMethodInvocation—Creation

```
WSMethodInvocationRef
WSMethodInvocationCreate(   CFURLRef url,
   CFStringRef methodName,  CFStringRef protocol)
```

- Protocols:

  kWSXMLRPCProtocol

  kWSSOAP1999Protocol

WSMethodInvocation—Parameters

void
WSMethodInvocationSetParameters(
WSMethodInvocation ref, CFDictionaryRef values,
CFArrayRef paramOrder)

- Serialization of parameters
  - CFBoolean, CFNumber, CFDate, CFString, CFDate, CFData, CFArray, CFDictionary
  - Complex types built using CFDictionary
  - Meta data keys can be added to specify custom namespaces, field ordering
  - Types are serialized using “Section 5” SOAP encoding

WSMethodInvocation—Parameters

CFDictionaryAddValue(dict, CFSTR(“param1”), CFSTR(“Steve”)

CFDictionaryAddValue(dict, CFSTR(“param2”), CFSTR(“John”))

CFArrayAddValue(order, CFSTR(“param1”))

CFArrayAddvalue(order, CFSTR(“param2”))

WSMethodInvocationSetParameters(ref, values, order)

WSMethodInvocation—Invoke!

CFDictionaryRef
WSMethodInvocationInvoke(WSMethodInvocation ref)

- Result is a dictionary (you must release)
- Optionally contains debug output of request, reply and HTTP headers
- SOAP headers returned as kWSSOAPHeaderValues CFArray of CFStringRef
- Faults may be manufactured for networking errors (kWSNetworkStreamFaultString)

WSMethodInvocation—Result Dictionary

- If . . .

  WSMethodResultIsFault(resultDict)

  CFTypeRef faultString = CFDictionaryGetValue(result, kWSFaultString)

  Also: **kWSFaultCode, kWSFaultExtra**

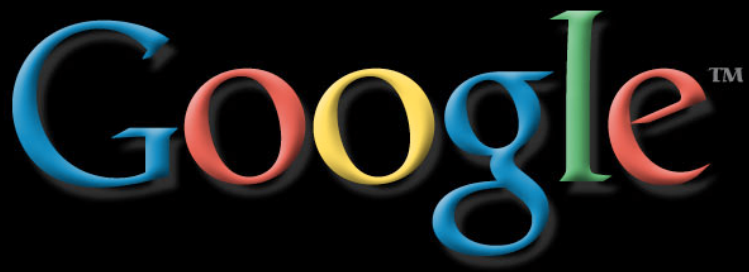
- Else, not a fault:

  CFTypeRef myResult = CFDictionaryGetValue(result, kWSMethodInvocationResult)

WSMethodInvocation—Asynchronous

- Set the client (callback)
- Schedule on a run loop
- Callback invoked from the runloop when the invocation completes
- Can be scheduled on multiple runloops to implement thread groups

Demo



SOAP Method Invocation
Accessing the Google API With SOAP

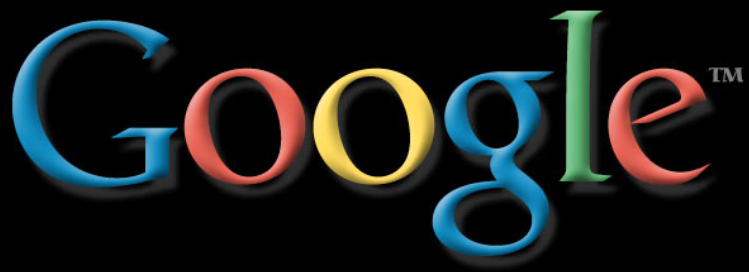
Customizations

- Serialization
  - **WSMethodInvocationAddSerializationOverride**
    - Based on the CFTypeID of the object being serialized
    - Result is a CFString containing an XML element
      <myNS:myType xmlns:myNS="myURI" magic="hat"/>
- Deserialization
  - **WSMethodInvocationAddDeserializationOverride**
    - Based on the namespace and type string from the source document
    - Access to the serialized type via CFXMLTree
- Headers/Extensions
  - Access the SOAP headers
  - SOAP extensions exist, not inherently supported

WSDL Support

- Web Services Description Language
- **/Developer/Tools/WSMakeStubs** to generate static stubs
- C++, ObjC, AppleScript
- Various degrees of complexity in generated stubs
- “Section 5” encoding and simple marshalling

Demo



WSMakeStubs
Accessing the Google API With SOAP

Future

- SOAP Extensions
  - Call out some soap extensions
- WSDL API
  - Must be true to itself
  - Full support for schemas -> types
- UDDI API
  - Service location
- Server Side API
  - WebObjects now!

Roadmap

805 Introducing CFNetwork	Room C **Tue., 5:00pm**
704 XML in WebObjects	Room A1 **Tue., 5:00pm**
405 Java Web Services	Room C **Wed., 10:30am**
808 Managing I/O: CFRunLoop and CFStream	Room C **Wed., 2:00pm**

Roadmap

705 WebObjects and Web Services	Room A1 **Wed., 2:00pm**
902 AppleScript Studio Intro	Civic **Wed., 3:30pm**

Who to Contact

Tom Weyer
Network and Communications Evangelist
weyer@apple.com

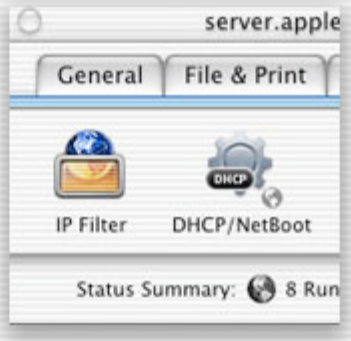
http://developer.apple.com/wwdc2002/urls.html

For More Information

Links

- Apple Sample Code
  /Developer/Examples/Web Services/
- XML-RPC
  www.xmlrpc.com
- SOAP
  www.w3c.org/TR/SOAP
- WSDL
  www.w3c.org/TR/WSDL
- UDDI
  www.uddi.org
- Adding OSA/AppleScript to applications (QA 1111)

Q&A



Tom Weyer
Network and Communications Evangelist
weyer@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002